

PROC SQL

When are leading and trailing blanks removed from macro variables?

Macro variables created with %LET have leading and trailing blanks removed.

```
%let num =      1234      ;  
%put NOTE: >>>&num<<< ;
```

Submitting this code generates the LOG:

```
1      %let num =      1234      ;  
2      %put NOTE: >>>&num<<< ;  
NOTE: >>>1234<<<
```

Using CALL SYMPUT with either an implicit or explicit conversion does not remove leading and trailing blanks.

```
data _null_ ;  
    call symput('num2',1234) ;  
    call symput('num3',put(1234,8.)) ;  
run ;  
  
%put NOTE: Converted values are right-aligned. ;  
%put NOTE- Implicit conversion uses BEST12. to convert: >>>&num2<<< ;  
%put NOTE- Explicit conversion determined by the 'w' value: >>>&num3<<< ;
```

Submitting this code generates the LOG:

```
1      data _null_ ;  
2          call symput('num2',1234) ;  
3          call symput('num3',put(1234,8.)) ;  
4      run ;
```

NOTE: Numeric values have been converted to character values at the places given by:

(Line):(Column).

2:22

NOTE: DATA statement used (Total process time):
real time 0.00 seconds

PROC SQL

cpu time

0.00 seconds

```
5
6   %put NOTE: Converted values are right-aligned. ;
NOTE: Converted values are right-aligned.
7   %put NOTE- Implicit conversion uses BEST12. to convert: >>>&num2<
<< ;
      Implicit conversion uses BEST12. to convert: >>>          1234<<<
8   %put NOTE- Explicit conversion determined by the 'w' value: >>>&n
um3<<< ;
      Explicit conversion determined by the 'w' value: >>>          1234<<<
```

CALL SYMPUTX automatically removes leading and trailing blanks.

```
data _null_ ;
  call symputx('num4',put(1234,8.)) ;
run ;

%put NOTE: >>>&num4<<< ;
```

Submitting this code generates the LOG:

```
1   data _null_ ;
2     call symputx('num4',put(1234,8.)) ;
3   run ;
```

```
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds
```

```
4
5   %put NOTE: >>>&num4<<< ;
NOTE: >>>1234<<<
```

The INTO clause in PROC SQL does not remove leading and trailing blanks either; use a %LET statement to resolve the issue.

```
proc sql noprint ;
```

PROC SQL

```
select count(*) into :num5
from sashelp.class
;
quit ;
```

```
%put NOTE: >>>&num5<<< ;
```

```
%let num5 = &num5 ;
%put NOTE: >>>&num5<<< ;
```

Submitting this code generates the LOG:

```
1  proc sql noprint ;
2      select count(*) into :num5
3      from sashelp.class
4      ;
5  quit ;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.01 seconds
      cpu time            0.00 seconds

6
7  %put NOTE: >>>&num5<<< ;
NOTE: >>>      19<<<
8
9  %let num5 = &num5 ;
10 %put NOTE: >>>&num5<<< ;
NOTE: >>>19<<<
```

Alternatively use the SEPARATED BY clause (even when expecting a single value) and leading and trailing blanks 'are' removed.

```
proc sql noprint ;
    select count(*) into :num6 separated by ''
    from sashelp.class
    ;
quit ;

%put NOTE: >>>&num6<<< ;
```

PROC SQL

Submitting this code generates the LOG:

```
1  proc sql noprint ;
2      select count(*) into :num6 separated by ''
3      from sashelp.class
4      ;
5  quit ;
```

```
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds
```

```
6
7  %put NOTE: >>>&num6<<< ;
NOTE: >>>19<<<
```

Unique solution ID: #1009

Author: Alan D Rudland

Last update: 2017-03-16 12:46